CENTRE DE RECHERCHE
SUR LES TRANSPORTS
(C.R.T.)
CENTRE FOR RESEARCH
ON TRANSPORTATION

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE

CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A,
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE: (514) 343-7575
TÉLEX: (21) 05-561325
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121

CRT-771

Sequence Dependent Scheduling to Minimize the Number of Late Jobs

by
Michel Gendreau[1]
Gilbert Laporte[1]
Marius M. Solomon[2]

[1] Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale A, Montréal, Canada H3C 3J7.

[2] College of Business Administration, Northeastern University, 314 Hayden Hall, Boston, MA 02115, U.S.A. Visiting professor at the Centre de recherche sur les transports and at the GERAD, Montreal.

Centre de recherche sur les transports — Publication #771
April 1991

91-04844/

Dépôt légal – 2ème trimestre 1991

ABSTRACT

In this paper the following job scheduling problem on a single processor is considered. There are n jobs j with deadlines d_j and processing times t_{ij}, if job j is executed immediately after job i. The objective is to minimize the number of jobs completed after their deadline. The problem can also be viewed in a vehicle routing context where t_{ij} represents the travel time from customer i to customer j. This problem is NP-hard, but a class of solvable cases can be characterized. An integer linear programming formulation is proposed, and the problem is solved by means of a specialized enumerative algorithm. Computational results are presented for problems involving up to 100 jobs.

Keywords: scheduling, sequencing, deadlines.

RÉSUMÉ

On considère dans cet article un problème d'ordonnancement de n tâches sur une machine. Plus précisément, chaque tâche j possède une échéance d_j, et la tâche j requiert un temps d'exécution t_{ij} si elle est exécutée immédiatement après la tâche i. L'objectif consiste à minimiser le nombre de tâches complétées après leur échéance. On peut aussi concevoir ce problème comme un problème de tournées de véhicules, où t_{ij} correspond au temps de déplacement du client i au client j. Le problème est NP-difficile, mais on peut néanmoins caractériser une classe de problèmes résolubles en temps polynomial. On formule le problème comme un programme linéaire en nombres entiers que l'on résout à l'aide d'un algorithme énumératif spécialisé. On présente des résultats numériques pour des problèmes comportant jusqu'à 100 tâches.

Mots-clés: ordonnancement, échéances.

1. Introduction

In this paper, we consider the following job scheduling problem on a single processor. There are n jobs j with deadlines d_j and processing times t_{ij}, if job j is executed immediately after job i; if job j is executed first, its processing time is t_{oj}. The objective is to minimize the number of late jobs, i.e., jobs completed after their deadline. This problem occurs naturally in production scheduling. It can also be viewed in a vehicle routing context. Here, job j corresponds to a delivery made to customer j, and t_{ij} is the travel time from i to j. All deliveries must be "processed" by the same uncapacitated vehicle.

The objective of minimizing the number of late jobs is encountered in several practical applications. In advanced manufacturing systems run on just-in-time principles, it is important to minimize the number of idle machines and thus the number of machines to which deliveries occur after they have finished processing the current part. Other examples include express pizza delivery systems, particularly in the case of companies competing on service differentiation rather than on cost minimization – see Hart and Delfino (1986).

The problem can be viewed as a generalization of the scheduling problem originally treated by Moore (1968), in which processing times are sequence independent and the same objective is minimized. Kise, Ibaraki and Mine (1978) later treated an extension in which deadlines d_j are replaced by time windows $[a_j, b_j]$. Similar problems involving sequence dependent processing times, but a different objective, have also been studied. The simplest one consists of minimizing the makespan, i.e.,

the finishing time of the last job (Bratley, Florian, and Robillard, 1971).

A related problem is the selective traveling salesman problem (STSP) recently treated by Laporte and Martello (1990). In this problem, each job j has a profit p_j and all jobs have a common deadline d. Again, processing times are sequence dependent. The STSP consists of maximizing the total profit of all jobs completed within time d.

In addition, there exists an extensive literature on scheduling problems involving independent processing times and time windows. This research is surveyed in several articles, e.g., Sen and Gupta (1984), Gupta and Kyparisis (1987), Cheng and Gupta (1989), and Baker and Scudder (1990). Similarly, research involving the minimization of makespan for models with sequence dependent processing times and time windows has been reviewed by Desrosiers and Solomon (1988).

We believe the problem of minimizing the number of late jobs in the presence of sequence dependent processing times and due dates has never been treated in the literature. Our objective is to provide an exact algorithm for this problem.

The remainder of the paper is organized as follows. In Section 2, we show that the problem is NP-hard and we examine a class of polynomially solvable cases. An integer linear programming formulation is presented in Section 3 and an enumerative algorithm is described in Section 4, followed by computational results in Section 5, and by conclusions in Section 6.

2. Complexity and solvable cases

The problem is NP-hard in the strong sense, as can be seen by transformation of its recognition version from the Hamiltonian circuit problem (HC), known to be strongly NP-complete (Garey and Johnson, 1979). Given any instance of the HC, relative to a graph $G = (V, A)$, with vertex set $V = \{v_1, ..., v_n\}$ and arc set $A = \{(v_i, v_j)\}$, define an instance of the problem having n jobs with common deadline $d_i = n$, $i = 1, ..., n$, $t_{ij} = 1$ if $(v_i, v_j) \in A$, $t_{ij} = 2$ if $(v_i, v_j) \notin A$. Then G possesses a Hamiltonian circuit if and only if the solution to the problem is equal to 0.

Nevertheless, there are some important solvable cases of this problem. For example, consider the model where after a possible permutation, the processing times satisfy $t_{ij} = t_{ik} + t_{kj}$, for $i < k < j$. In this case, the identity permutation is optimal. This is so because the minimum flow time after completion of job i is $\sum_{r=2}^{i} t_{r-1,\,r}$, and switching jobs i and j would increase the flow time after job j. The model has important applications in routing where all customers are located on a line or on a circle. Advanced manufacturing systems often exhibit this kind of structure for their automated guided vehicles system – see Jaikumar and Solomon (1990), and Blazewicz, Eiselt, Finke, Laporte and Weglarz (1991).

Another important solvable case is the so-called *shoreline problem*, a routing structure recently analyzed by Psaraftis, Solomon, Magnanti and Kim (1990). In this problem, the n points to be visited are located along a continuous line that does not cross itself, and all trips must be made on one side of the line only, as on a

shoreline. More formally, the shoreline distance (travel time) structure must satisfy the following conditions for all $1 \leq i \leq k \leq j \leq n$:

(i) $t_{ii} = 0$,

(ii) $t_{ij} = t_{ji}$,

(iii) $t_{ij} \geq t_{ik}$,

(iv) $t_{ij} \geq t_{kj}$,

(v) $t_{ij} \leq t_{ik} + t_{kj}$.

It is possible to check in $O(n^2)$ time whether any given distance matrix satisfies these conditions. In the absence of release time and deadline constraints, the shoreline problem with makespan objective is polynomially solvable by traversing the points in order from 1 to n. Consider now the makespan shoreline problem with deadline constraints satisfying "shoreline rules", i.e., $i \leq j$ implies that $d_i \leq d_j$. The solution to this problem can also be obtained in polynomial time.

Theorem 1. The optimal solution to the shoreline problem with shoreline deadlines for the makespan objective is the identity permutation, if this is feasible.

Proof: Suppose that permutation π is feasible, but that the identify permutation is not. Let k^* be the index of the first job which is not feasible under the identity permutation, i.e. such that its completion time c_{k^*} exceeds d_{k^*}. Let r^* be the position of k^* in π, k' the last processed job in $\{1, ..., k^*\}$ under π, and r' its position in π. Note that $r' \geq r^*$. Consider now the set of jobs $J = \{\pi(1), ..., \pi(r')\}$, where $\pi(i)$ denotes the job in position i of π. The identity permutation of these jobs has a completion time

no larger than $c_{\pi(r')}$. Since J contains $\{1, ..., k^*\}$, this implies

$$\begin{aligned} c_{k^*} &\leq c_{\pi(r')} \\ &\leq d_{k'} \quad \text{(since } \pi \text{ is feasible)} \\ &\leq d_{k^*} \quad \text{(by the shoreline assumption on due dates),} \end{aligned} \tag{1}$$

which contradicts the assumption that the identity permutation is not feasible. Furthermore, since the identity permutation is optimal for the problem without deadline constraints, if it is feasible with respect to these constraints, it must also be optimal. It follows that if the identity permutation is feasible, it is also optimal. □

The shoreline problem with makespan objective can therefore be solved in $O(n)$ time. The following theorem shows that if the objective is to minimize the number of late jobs, the shoreline problem can still be solved in polynomial time.

Theorem 2. There exists an optimal solution to the shoreline problem with shoreline deadlines with the objective of minimizing the number of tardy jobs in which all jobs completed on time are processed "in order".

Proof: Consider any optimal solution. Let J be the set of jobs that are processed on time for this solution. Applying Theorem 1 to J implies that processing these jobs in order meets all deadlines and yields an optimal completion time for these jobs. □

The optimal schedule can be found by solving the following dynamic program. Let $f(j, r)$ be the optimal completion time of job j if r jobs are skipped before j; if job j cannot be completed before its deadline then $f(j, r) = \infty$. For $j = 1, ..., n$ and $r =$

$0, ..., j-1$,

$$f(j,r) = \begin{cases} g(j,r) & \text{if } g(j,r) \leq d_j \\ \infty & \text{otherwise,} \end{cases}$$

where

$$g(j,r) = \min_{j-r-1 \leq i \leq j-1} [t_{ij} + f(i, r-(j-i-1))] \text{ if } r < j-1$$

and

$$g(j,j-1) = t_{oj}.$$

Job $i^*(j,r)$ is the one which precedes j in the optimal sequence $(0,1,...,j)$ with r jobs skipped. The boundary conditions are:

$$f(n+1,r) = \min_{n-r \leq i \leq n} \{f(i, r-(n-i))\} \qquad (r = 0, ..., n-1)$$

$$f(n+1,n) = 0.$$

Here, $f(n+1,r)$ is the optimal completion time of a sequence with r skipped jobs ($f(n+1,r) = \infty$ if there is no feasible sequence with r skipped jobs). Then $r^* = \min\{r : 0 \leq r \leq n \text{ and } f(n+1,r) < \infty\}$ is the minimum number of late jobs. The dynamic program can be solved in $O(n^3)$ operations. In practice, however, one could use a reaching algorithm (Denardo 1982, p. 16).

3. Formulation

The problem can be formulated as an integer linear program as follows. Let u_j be a feasible finishing time for job j if it is completed before its deadline; otherwise, u_j has no particular meaning and is forced to zero in the solution. Let x_{ij} be a binary variable equal to 1 if and only if job j is scheduled immediately after job i. Note that all jobs need not be scheduled and that x_{ij} must be interpreted as 0 if $i = j$. Job

0 is a dummy job used as the process starting point. Hence $\sum_{i=0}^{n} x_{ij} = 1$ if and only if job j is scheduled. The formulation is then

$$\text{minimize} \quad \sum_{j=1}^{n} \left(1 - \sum_{i=0}^{n} x_{ij}\right)$$

subject to

$$\sum_{k=1}^{n} x_{jk} \leq \sum_{i=0}^{n} x_{ij} \leq 1 \qquad (j = 1, ..., n) \qquad (2)$$

$$u_j \geq t_{0j} - T(1 - x_{0j}) \qquad (j = 1, ..., n) \qquad (3)$$

$$u_j \geq u_i + t_{ij} - T(1 - x_{ij}) \qquad (i, j = 1, ..., n) \qquad (4)$$

$$0 \leq u_j \leq d_j \sum_{i=0}^{n} x_{ij} \qquad (j = 1, ..., n) \qquad (5)$$

$$x_{ij} \in \{0, 1\} \qquad (i = 0, ..., n; j = 1, ..., n), \qquad (6)$$

where T is an arbitrarily large number.

In this formulation, constraints (2) ensure that job j can have a successor only if it has a predecessor, and that the number of jobs immediately preceding or immediately following job j cannot exceed 1. Constraints (3) and (4) imply that if job j immediately follows job i, then $u_j \geq u_i + t_{ij}$. Constraints (4) also prevent the formation of "subtours", i.e. job sequences of the form $(j_1, ..., j_s, j_1)$ (see Solomon and Desrosiers, 1988). Moreover, as in Desrochers and Laporte (1991), applying a lifting process to constraints (4) yields the stronger constraints:

$$u_j \geq u_i + t_{ij} - T(1 - x_{ij}) + (T - t_{ij} - t_{ji})x_{ji} \quad (i, j = 1, ..., n). \qquad (4')$$

Constraints (4') are valid since setting $x_{ij} = 1$ implies $x_{ji} = 0$ and $u_j \geq u_i + t_{ij}$; similarly, setting $x_{ji} = 1$ implies $x_{ji} = 0$ and $u_i \leq u_j + t_{ji}$; when $x_{ij} = x_{ji} = 0$, constraints (6') are ineffective. Constraints (5) mean that if job j is scheduled, its finishing time cannot exceed its deadline.

The value of this model is mostly descriptive since the problem under consideration is comparable to a TSP with time windows for which similar formulations are not easily amenable to exact solution by means of existing integer programming methodology. In the following section, we describe a specialized algorithm for the problem.

4. Enumerative algorithm

We now present a depth-first branch-and-bound procedure for the problem. The algorithm gradually extends a feasible sequence of jobs $(j_1, j_2, ..., j_\ell)$ until no more jobs can be added, or the current solution is dominated. Lower bounds on the value of the optimum are computed at each node, applying a procedure due to Hodgson, and described in Moore (1970). Moreover, using an argument presented in Fischetti and Toth (1988), we apply a dominance test to fathom some nodes of the search tree.

We first describe Hodgson's procedure. Given a process starting at time c, a subset R of jobs with deadline vector (d_j) and processing time vector (t_j), compute m, the minimum number of missed deadlines. An optimal solution to this problem is obtained by executing the following operations.

PROCEDURE HODGSON $(c, R, (d_j), (t_j), m)$

Step 1. Set $m := 0$

Step 2. Consider in turn all jobs $j \in R$, in non-decreasing order of deadline; schedule job j and set $c := c + t_j$; if $c > d_j$, identify from the set S all scheduled jobs a job j^* satisfying $t_{j^*} = \max_{j \in S} \{t_j\}$, set $c := c - t_{j^*}$, $m := m + 1$, and remove j^* from S. □

In order to describe the main algorithm, the following notation will be required:

$J = \{1, ..., n\}$: the set of all jobs, given in a non-decreasing order of deadlines;

z^*: the value of the best known solution.

For every node h of the search tree, define:

$\underline{z}_h$: a lower bound on z^*;

$\overline{z}_h$: an upper bound on z^*;

ℓ_h: the last job in the sequence of scheduled jobs;

c_h: the completion time of the last job ℓ_h;

S_h: the set of scheduled jobs;

R_h: the set of "reachable jobs", i.e. all jobs that can feasibly be scheduled immediately after ℓ_h.

The algorithm is then:

ALGORITHM DEADLINES

Step 0. (Initialization). Set $\ell_0 := 0$, $c_0 := 0$, $S_0 := \emptyset$, $z^* := n$, the best solution is empty. Insert node 0 in a stack.

Step 1. (Termination check). If the stack is empty, print the optimal solution and stop. Otherwise, remove from the stack the last created node. Let h be this node.

Step 2. (Lower bound computation). Set $R_h := \{j \in J \setminus S_h : c_h + t_{\ell_h j} \leq d_j\}$. If $R_h = \emptyset$, go to Step 1. Otherwise, for all $j \in R_h$, compute $t_j = \min_{i \in \{\ell_h\} \cup R_h} \{t_{ij}\}$. Compute $\Delta := \min_{j \in R_h} \{t_{\ell_h j} - t_j\}$. For all $j \in R_h$, set $\overline{d}_j := d_j - \Delta$. Perform PROCEDURE HODGSON $(c_h, R_h, (\overline{d}_j), (t_j), m_h)$ and set $\underline{z}_h := m_h + (n - \mid S_h \mid - \mid R_h \mid)$. If $\underline{z}_h \geq z^*$, go to Step 1.

Step 3. (Updated incumbent). Consider the sequence of scheduled jobs provided by HODGSON in Step 2, and compute the number m'_h of missed deadlines obtained by using the original processing times t_{ij}. This provides a feasible solution with cost $\overline{z}_h = m'_h + (n - \mid S_h \mid - \mid R_h \mid)$. If $\overline{z}_h < z^*$, set $z^* := \overline{z}_h$ and update the best solution. If $\underline{z}_h = \overline{z}_h$, go to Step 1.

Step 4. (Fathoming procedure). If $\mid S_h \mid \geq 3$, check whether a smaller value of c_h can be obtained by successively attempting to move the job preceding ℓ_h in all possible positions. As soon as it is discovered that a move decreases c_h, go to Step 1.

Step 5. (Node creation). Create a new node h_j for every $j \in R_h$ by inserting j after ℓ_h. In each case, set $\ell_{h_j} := j$, $c_{h_j} := c_h + t_{\ell_h j}$ and $S_{h_j} := S_h \cup \{j\}$. Insert these nodes in the stack and go to Step 1. □

Here, two remarks are in order. First, in Step 2, t_j represents the minimum time to reach any of the jobs of R_h. But since one of these jobs must be executed immediately after ℓ_h, Δ additional time units will be required, and it is therefore valid to reduce all deadlines by Δ in HODGSON. Secondly, in Step 4, node h can

be fathomed if there exists a permutation of the current sequence of jobs that leaves the last job unchanged, but yields a shorter completion time c_h. This is so since the permuted schedules are considered elsewhere in the search tree. This dominance argument was suggested by Fischetti and Toth (1988) in a more general context.

5. Computational results

The algorithm was tested on a large number of instances. These were derived from benchmark problems proposed by Solomon (1987) to compare algorithms for the vehicle routing problems with time windows. We used as processing times the interpoint distances of these problems, and as deadlines, the upper limit of the time windows. These problems exhibit three basic types of distance structures; (i) random distribution of points in the plane; (ii) clustered points, and (iii) a combination of these two features. We used 6 random problems (R101, R105, R109 – R112), 5 clustered problems (C101, C105, C107–C109), and 5 random clustered problems (RC101, RC105–RC108). For each data set we used the first n points, and for each value of n, ten versions of the problem were obtained by multiplying the deadlines by a factor $\alpha = 0, 0.1, 0.2, ..., 1$, so as to create a large number of problems ranging from highly constrained and loosely constrained instances. All problems were solved on a Sun Sparc Workstation. Computational results are reported in Table 1. The column headings as they appear from left to right are as follows:

n: number of jobs;

α: deadline coefficient;

UB: objective value of the first feasible solution encountered in the search tree;

LB: lower bound on z^* computed at the root of the search tree;

z^*: the optimum;

NODES: number of nodes in the search tree;

TIME: CPU time in seconds.

All results are average values over the number of basic problems (5 or 6) solved for each combination of n and α within a time frame of 1000 seconds per problem. When not all instances could be solved to completion, the number of "successful" problems is shown in brackets in the TIME column.

The results indicate that depending on the type of problem considered and on the value of α, the proposed algorithm can solve relatively large problems with up to $n = 100$. This is larger than the size of problems solved optimally when a makespan objective is considered. Random and clustered problems exhibit totally opposite behaviors. In the random case, interpoint distances tend to be larger on the average than in the case of clustered problems. As a result, when α is small, the number of feasible solutions is limited, resulting in a small search tree. In the case of clustered problems, the combination of problems with small $t_{ij}'s$ and a large value of α produces problems for which all or almost all jobs can be scheduled. Since the optimum is small, it is discovered at a low level of the search tree. As α decreases, z^* increases and a more extensive search becomes necessary. Random-clustered problems are easier than random problems, but these two classes exhibit basically the same behavior.

Table 1. Summary of computational results

		RANDOM PROBLEMS (6)					CLUSTERED PROBLEMS (5)					RANDOM-CLUSTERED PROBLEMS (5)				
n	α	UB	LB	z^*	NODES	TIME	UB	LB	z^*	NODES	TIME	UB	LB	z^*	NODES	TIME
20	1	10	2.5	6.5	133089	332.5(2)	0	0	0	1	0.01					
	.9	9.3	3	7.3	103861	262.6(3)	0	0	0	1	0.01	11	0	5	241400	892(1)
	.8	10.8	4.2	7.8	119379	303.2(5)	0	0	0	1	0.01	9.5	0	5	150409	400(2)
	.7	11.8	5.6	9	48572	115.5	0	0	0	1	0.01	10.2	0	6.2	134796	381(4)
	.6	12.1	7.5	10.3	12857	27.0	0	0	0	1	0.01	9.6	0	7.8	41789	100
	.5	13.3	8.8	11.6	2988	5.6	0	0	0	1	0.01	12.6	1.4	10.4	4650	11.4
	.4	14.8	10.6	13	703	1.0	0.5	0	0	2.5	0.03	12.8	5.4	11	764	2
	.3	16.1	13	14.6	145	0.2	2.6	0.2	0.2	5.4	0.04	15	12	14.2	128	0.2
	.2	16.8	15.3	16.5	23	0.03	5	0.7	0.7	18716	154	18	18	18	1	0.01
	.1	17.5	17.1	17.5	2	0.01	7.6	3.2	5.6	93384	290	19	19	19	1	0.01
40	1						3.2	0	0	11	0.6(4)					
	.9						4.3	0	0	12	0.7(3)					
	.8						6.3	0	0	22	1(3)	29	1	26	225751	750(1)
	.7	31	22	28	269504	743(1)	7	0	0	23	1(1)	30.5	2.5	26.5	110140	370.5(2)
	.6	31	23.6	28.6	170015	474(3)	11	0	0	25	1(1)	28.2	5.6	26.2	90204	315
	.5	32.3	25.6	29.3	130944	373	12	0	0	27	1(1)	29	4.8	28.2	8609	34.8
	.4	32.8	27.8	30.8	14405	36.8						32.4	20.2	30.8	1273	4.2
	.3	34	30.8	32.6	1485	3.3						33.2	26.8	32.6	201	0.5
	.2	35.6	33.6	35	134	0.2						37.8	37	37.8	3	0.01
	.1	37	33.6	37	3	0.01						39	39	39	1	0.01
60	1						17	0	0	35	6.5(1)					
	.6											50	15	47	165816	783(1)
	.5											51.3	22	48	41555	192.3(3)
	.4	51.6	43.6	49.3	89316	324.3(3)						46.8	15	47	32868	176
	.3	52.6	43.6	51	20467	68						48.8	22	48	1909	7.1
	.2	54.6	50.1	53.8	745	2						57.6	28.2	45.8	12	0.03
	.1	56.8	50.1	56	20	0.05						58	38.4	48.2	1	0.01
80	.4											68.5	49.2	66.2	68468	355.5(4)
	.3	72.4	62.4	69	76753	333.2(5)						68.4	58.4	68.4	5279	20.8
	.2	74.6	68	72.5	1917	6.9						76.2	73	75	77	0.2
	.1	76.8	74.8	76	23	0.1						77.6	77	77	2	0.01
100	.4											91	71	88	168394(1)	868.2
	.3	91	78	87	116369(2)	629						89.6	76.2	89.4	38925	194.2
	.2	92.8	84.3	90.5	18748	103.3						92.4	90.4	91.6	547	2.6
	.1	96.6	94.1	96	31	0.1						97	95.4	96	17	0.05

6. Conclusion

The scheduling problem treated in this paper occurs naturally in a variety of manufacturing and distribution contexts. It is NP-hard, but some polynomially solvable cases exist, such as problems exhibiting a shoreline structure. We have developed an enumerative algorithm for the problem. It includes a lower bounding procedure suggested by Hodgson for problems with sequence independent processing times. Instances involving up to 100 jobs have been solved optimally by means of the proposed approach. The difficulty of the problem is largely contingent upon the structure underlying the processing time matrix.

Acknowledgements

This research was partially supported by the Canadian Natural Sciences and Engineering Research Council (grants OGP0038816 and OGP0039682). Thanks are also due to Mihnea Stan for his help with programming.

References

BAKER, K.R. and G.D. SCUDDER, "Sequencing with Earliness and Tardiness Penalties", *Operations Research*, 38 (1990), 22–36.

BLAZEWICZ, J., EISELT, H.A., FINKE, G., LAPORTE, G. and J. WEGLARZ, "Scheduling Tasks and Vehicles in a Flexible Manufacturing System", *Flexible Manufacturing Systems*, (1991), forthcoming.

BRATLEY, P., FLORIAN, M. and P. ROBILLARD, "Scheduling with Earliest Start and Due Date Constraints", *Naval Research Logistics Quarterly*, 18 (1971), 511–519.

CHENG, T.C.E. and M. GUPTA, "Survey of Scheduling Research Involving Due Date Determination Decisions", *European Journal of Operations Research*, 38 (1989), 156–166.

DENARDO, E.V., "*Dynamic Programming-Models and Applications*", Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

DESROCHERS, M. and G. LAPORTE, "Improvements and Extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints", *Operations Research Letters*, 10 (1991), 27–36.

GAREY, M.R. and D.S. JOHNSON, "Computer and Intractibility: A Guide to the Theory of NP-Completeness", Freeman, San Francisco, 1979.

GUPTA, S. and S. KYPARISIS, "Single Machine Scheduling Research", *Omega* 15 (1987), 277–282.

HART, C. and M. DELFINO, "Caruso's Pizza: Express Delivery (B)", Case 9–687–025, Harvard Business School, 1986.

JAIKUMAR, R. and M.M. SOLOMON, "Dynamic Scheduling of Automated Guided Vehicles", *Journal of Manufacturing Systems*, 9 (1990), 315–323.

KISE, H., IBARAKI, T. and H. MINE, "A Solvable Case of the One-Machine Scheduling Problem with Ready and Due Times", *Operations Research*, 26 (1978), 121–126.

LAPORTE, G. and S. MARTELLO, "The Selective Travelling Salesman Problem", *Discrete Applied Mathematics*, 26 (1990), 193–207.

MOORE, J.M., "An n Job, one Machine Sequencing Algorithm for Minimizing the Number of Late Jobs", *Management Science*, 14 (1968), 102–109.

PSARAFTIS, H.N. , SOLOMON, M.M., MAGNANTI, T.L. and T.-U. KIM, "Routing and Scheduling on the Shoreline with Release Times", *Management Science*, 36 (1990), 212–223.

SEN, T., and S. GUPTA, "A State-of-the-Art Survey of Static Scheduling Research Involving Due Dates", *Omega*, 12 (1984), 62–74.

SOLOMON, M.M., "Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints", *Operations Research*, 35 (1987), 254–265.

SOLOMON, M.M. and J. DESROSIERS, "Time Window Constrained Routing and Scheduling Problems: A Survey", *Transportation Science*, 22 (1988), 1–13.